segment_liftover : a Python tool to convert segments between genome assemblies

Gao, Bo ; Huang, Qingyao ; Baudis, Michael

Abstract: The process of assembling a species' reference genome may be performed in a number of iterations, with subsequent genome assemblies differing in the coordinates of mapped elements. The conversion of genome coordinates between different assemblies is required for many integrative and comparative studies. While currently a number of bioinformatics tools are available to accomplish this task, most of them are tailored towards the conversion of single genome coordinates. When converting the boundary positions of segments spanning larger genome regions, segments may be mapped into smaller sub-segments if the original segment's continuity is disrupted in the target assembly. Such a conversion may lead to a relevant degree of data loss in some circumstances such as copy number variation (CNV) analysis, where the quantitative representation of a genomic region takes precedence over base-specific accuracy. segment_liftover aims at continuity-preserving remapping of genome segments between assemblies and provides features such as approximate locus conversion, automated batch processing and comprehensive logging to facilitate processing of datasets containing large numbers of structural genome variation data.

DOI: https://doi.org/10.12688/f1000research.14148.1

SOFTWARE TOOL ARTICLE

REVISED segment_liftover : a Python tool to convert segments between genome assemblies [version 2; referees: 2 approved]

Bo Gao [1,2], Qingyao Huang[1,2], Michael Baudis [1,2]

[1]Institute of molecular Life Sciences, University of Zürich, Zürich, CH-8057, Switzerland
[2]Swiss Institute of Bioinformatics, University of Zürich, Zürich, CH-8057, Switzerland

v2 **First published:** 14 Mar 2018, **7**:319 (doi: 10.12688/f1000research.14148.1)
Latest published: 08 Jun 2018, **7**:319 (doi: 10.12688/f1000research.14148.2)

Abstract

The process of assembling a species' reference genome may be performed in a number of iterations, with subsequent genome assemblies differing in the coordinates of mapped elements. The conversion of genome coordinates between different assemblies is required for many integrative and comparative studies. While currently a number of bioinformatics tools are available to accomplish this task, most of them are tailored towards the conversion of single genome coordinates. When converting the boundary positions of segments spanning larger genome regions, segments may be mapped into smaller sub-segments if the original segment's continuity is disrupted in the target assembly. Such a conversion may lead to a relevant degree of data loss in some circumstances such as copy number variation (CNV) analysis, where the quantitative representation of a genomic region takes precedence over base-specific accuracy. *segment_liftover* aims at continuity-preserving remapping of genome segments between assemblies and provides features such as approximate locus conversion, automated batch processing and comprehensive logging to facilitate processing of datasets containing large numbers of structural genome variation data.

Keywords

Genome assembly, liftover, remap, copy number segment.

Open Peer Review

Referee Status: ✓ ✓

	Invited Referees 1	2
REVISED version 2 published 08 Jun 2018		✓ report
version 1 published 14 Mar 2018	✓ report	? report

1 **Oscar Krijgsman**, Netherlands Cancer Institute, Netherlands

2 **Ryan K. Dale**, National Institutes of Health, USA

Discuss this article

Comments (0)

Corresponding author: Michael Baudis (mbaudis@me.com)

Author roles: Gao B: Methodology, Software, Writing – Original Draft Preparation; **Huang Q**: Formal Analysis, Visualization, Writing – Original Draft Preparation; **Baudis M**: Conceptualization, Supervision, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

How to cite this article: Gao B, Huang Q and Baudis M. **segment_liftover : a Python tool to convert segments between genome assemblies [version 2; referees: 2 approved]** *F1000Research* 2018, **7**:319 (doi: 10.12688/f1000research.14148.2)

Grant information: BG is recipient of a grant from the China Scholarship Council (CSC 201606620032). QH is supported throughthe University of Zurich's "CanDoc" program.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

First published: 14 Mar 2018, **7**:319 (doi: 10.12688/f1000research.14148.1)

REVISED **Amendments from Version 1**

In response to review comments: the logging and temp files mechanism is modified; new options are added to provide demonstration and more flexibility; the conversion result (previous figure 3) is replaced with a table, for clarity; adding more explanation for use-case 2 (comparison of different strategies) to elaborate the settings and purpose.

See referee reports

Introduction

The first draft version of human genome was published in 2001[1]. In subsequent years, several new editions were released to perfect the quality of the genome assembly. The current version of human genome (GRCh38, UCSC hg38) was made available in 2013, with the latest revision (Grch38.p12) still containing more than 10 million unplaced bases (see NCBI website). Over the years, large numbers of genomic studies have been performed, generating data mapped to different versions of the reference genome. However, when performing genome analyses integrating data from multiple resources, it is imperative to convert all data to the same genomic coordinate system.

Two general methodologies are used for conversion between coordinates from different genome assemblies. The first approach is to re-align the original sequence data to the target assembly. This method could provide the best result but is very time-consuming, and is not possible if the original sequence data is not available or does not consist of direct sequences (i.e. segmentation of array based data). Another approach is to convert the coordinates of genome data between assemblies by using a mapping file. This method, although bearing a side effect of minor information loss, for most applications provides a good balance between performance and accuracy.

Currently, three tools are in widespread use for the conversion between genome assemblies by coordinates: *liftOver* from University of California, Santa Cruz (referred as *UCSC liftOver* in the following article)[2]; *CrossMap* from Zhao[3]; and *Remap* from NCBI[4]. The *UCSC liftOver* tool exists in two flavours, both as web service and command line utility. It offers the most comprehensive selection of assemblies for different organisms with the capability to convert between many of them. *CrossMap* has the unique functionality to convert files in BAM/SAM or BigWig format. It generates almost identical results as *UCSC liftOver*, but is not optimised for converting genome coordinates between species. *Remap* provides for each organism a comprehensive list of major assemblies and the corresponding subversions. It can also perform cross species mapping, however, with only a limited number of organisms.

All those tools are efficient in coordinate conversion and provide almost identical results. However, as shown in Figure 1a, challenges arise when dealing with genome segments that are not continuous anymore in the target assembly; there, these three tools take on different strategies. *CrossMap* and *UCSC liftOver* break the segment into smaller segments and map them to different locations. *Remap* keeps the integrity of the segment and maps the span to the target assembly.

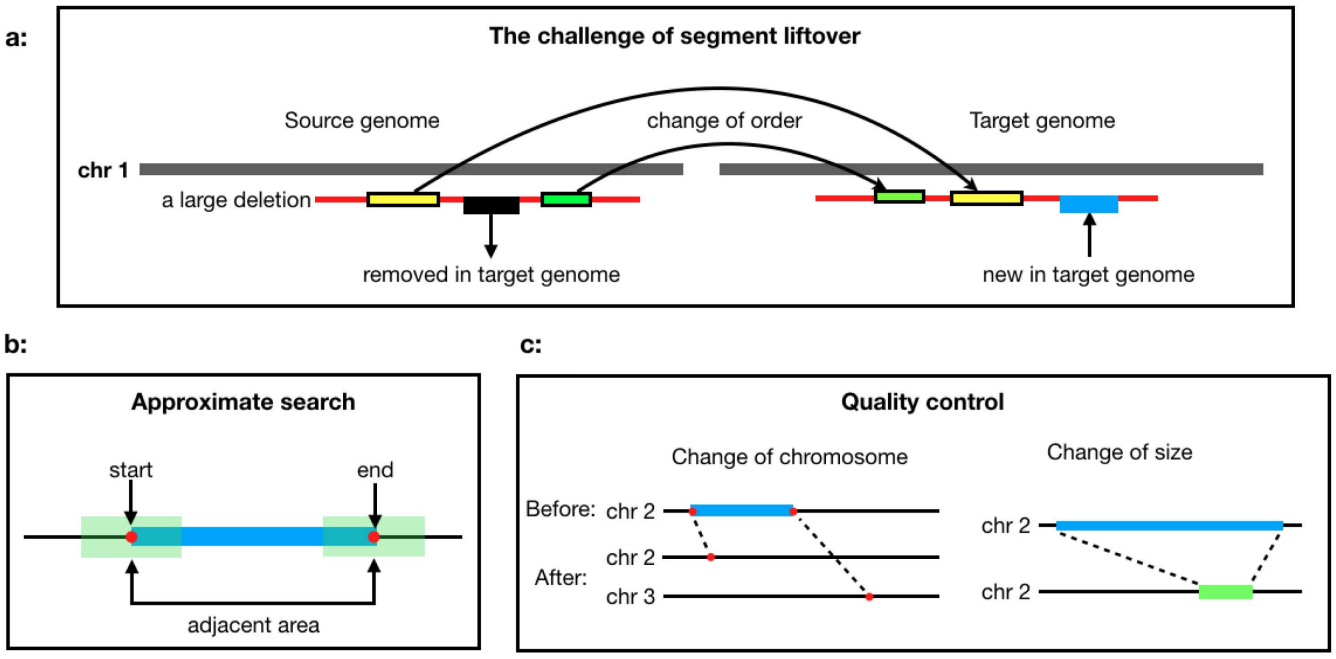

Figure 1. The challenge of segment liftover: (**a**) When lifting a segment to another assembly, the landscape of the segment may be affected by indels and copy number variations, but the overall span of the segment does not change significantly. (**b**) When the end positions cannot be converted by the *UCSC liftOver*, the nearby regions will be searched for convertible positions as approximation. (**c**) Quality control checks for changes of chromosome or size to make sure the segment is converted properly.

In research such as analysis of copy number variation (CNV) data, where the quantitative representation of a genomic range takes precedence over base-specific representation, the integrity of a continuous segment indicates the proper conversion between assemblies, but may not be a direct outcome of current re-mapping approaches. Although *Remap* can convert contiguous segments, it only provides web service with submission limits, which is difficult to use for large scale or pipelined applications. The limitation to single input files is a general limitation of those tools, which precludes their direct use in comparative studies which may require to work with hundreds or thousands of files, as indicated through our own projects and requests from the research community.

In this article, we introduce *segment_liftover*, a tool to perform an integrity-preserving conversion of genomic segments data between genome assemblies. It features two major functional additions over existing tools: First, re-conversion by locus approximation, in instances where a precise conversion of genomic positions fails; and second, the capability to handle any number of files and optional integration into data processing pipelines with supporting features such as automatic file traversal, interruption resumption and detailed logging.

Methods

Implementation

segment_liftover can convert both probe files and segment files at the same time or in separate runs. It starts from a structured directory or a list of files, then traverses and converts all files meeting the specified name pattern, and finally outputs to a designated directory. To convert a probe file, *segment_liftover* will first use the *UCSC liftOver* to convert the file, then apply an approximate conversion on probes that the *UCSC liftOver* failed to convert. To convert a segment file, *segment_liftover* will use the *UCSC liftOver* to convert the *start* and *end* positions of segments. A successful segment conversion needs to satisfy the following four criteria:

$$position_{new\_start} \neq \emptyset \quad (1)$$

$$position_{new\_end} \neq \emptyset \quad (2)$$

$$chromosome_{new\_start} = chromosome_{new\_end} \quad (3)$$

$$\frac{1}{\beta} < \frac{length(old\_segment)}{length(new\_segment)} < \beta \quad (4)$$

Where β controls the threshold of the length ratio and the default value is set to 2. If criteria (1) or (2) fails, *segment_liftover* will apply an approximate conversion; if the conversion still fails, it is reported as unconvertible. If criteria (3) or (4) fails, the conversion is reported as rejected (Figure 1c). The reason of failure is recorded in log files.

When a position cannot be converted by the *UCSC liftOver*, *segment_liftover* will attempt an approximate conversion and try to find a convertible position in the adjacency (Figure 1b). The range and the resolution of the search is defined by parameters *–range* and *–step_size*, respectively.

Operation

The *segment_liftover* tool is implemented in Python. The package is available for both Linux and OSX. It requires a minimum of 2G memory and the capacity of running Python 3. We recommend an installation using *pip* in a Python virtual environment. *segment_liftover* requires and depends on the *UCSC liftOver* program. A chain file, which provides alignments from source to target assembly, is also required. The chain files between common human assemblies (hg18, hg19 and hg38) are included in the program package. Chain files of other species and assemblies are available from the USCS Genome Browser. Figure 2 illustrates the work-flow of *segment_liftover*.

Use cases

In this section, we provide two examples of using *segment_liftover* to convert probes and segments, respectively. The two examples are part of the pipeline which updates the *arrayMap* database, a reference resource of somatic genome copy number variations in cancer[5], from human genome assembly hg19 to hg38.

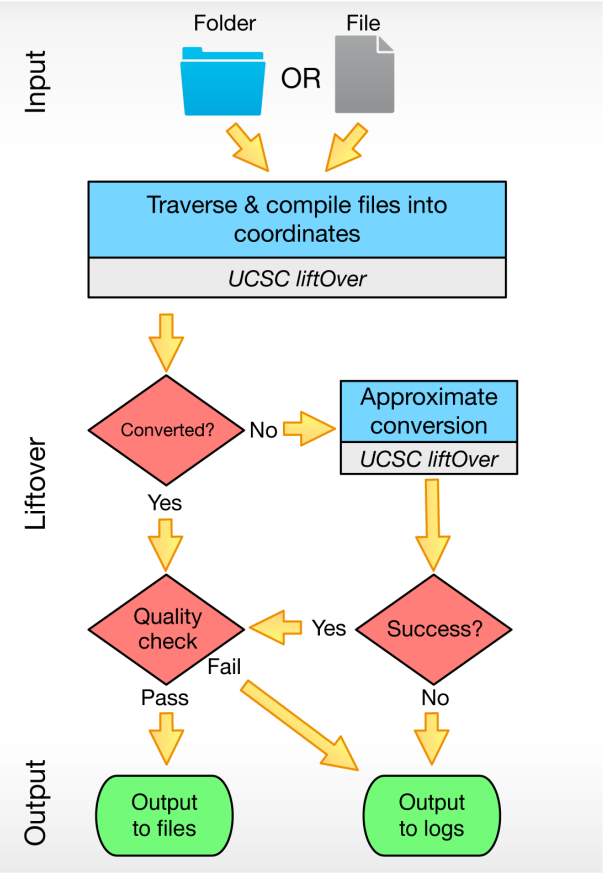

Figure 2. The workflow of *segment_liftover*: (1) It can take either a folder or a file containing the list of files as the input. (2) It will try to convert by approximation when *UCSC liftOver* fails to convert a coordinate. (3) The directory structure will be kept in the output folder and detailed log files are also available.

Converting arrayMap data from hg19 to hg38

In the first example, we converted 44,632 probe files and 44,471 segment files from hg19 to hg38. The probe data were generated from nine Affymetrix genotyping array platforms, which currently only support annotations for hg19. Circular binary segmentation (CBS) analysis (DNAcopy R-package) was used to infer copy number segments from log2 values of probes. The final segment files contain a list of genomic regions separated by their copy number values[6]. We ran the *segment_liftover* tool on a 12-core, 128GB RAM machine with 8 parallel processes. It took 42 hours to convert 44,632 probe files with 5.5 billion probe positions and 40 minutes to convert 44,471 segment files with 4.8 million segments.

Overall, more than 99.99% of probes and more than 99% of segments could be directly converted from hg19 to hg38 (Table 1). As conversion of segments is more complicated and involves a quality control procedure to ensure the meaningfulness of the segment, it is expected to have a higher number of unconvertible segments than probes. As shown in Figure 3, the unconvertible regions are mainly around telomeres, centromeres, or other gene-sparse locations. In total, 38 genetic elements were found to be affected by the conversion (description in Supplementary Table 1).

Comparison of different conversion strategies

In the second example, we compared the performance of different conversion strategies using 1,000 samples randomly drawn from the first example (Table 2). Copy number segments were generated using four different strategies (Figure 4): (1) segments in hg19, which were generated directly from probes in hg19; (2) segments in hg38, which were generated from probes converted from hg19 to hg38 using *segment_liftover*;

Table 1. Conversion results: *Directly converted* is the sum of successful conversion from *UCSC liftOver*; *approximately converted* is the sum of successful approximate conversion; *converted but rejected* is the sum of all rejections from quality control; *unconvertible* is the sum of everything that can not be converted at all.

	probes	segments
directly converted	5,515,432,156	4,800,078
approximately converted	373,788	14,707
converted but rejected	953,624	41,201
unconvertible	10,914	1,064

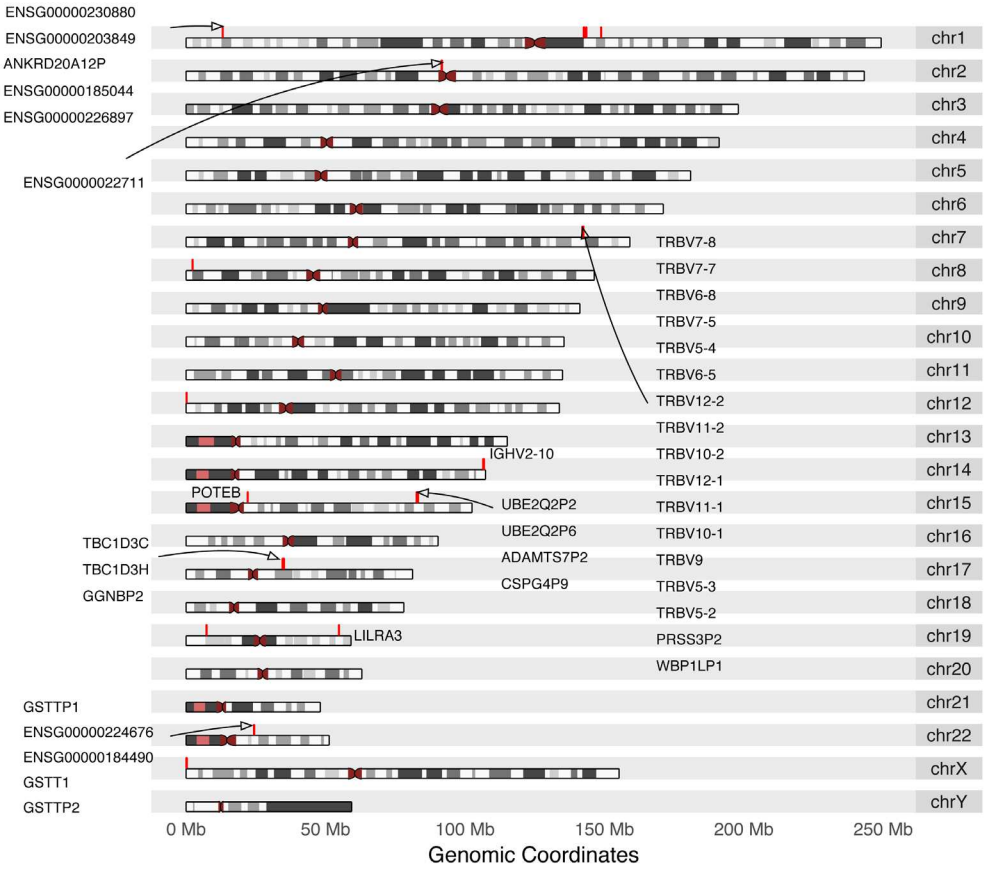

Figure 3. Genomic regions of unconvertible probe positions from human genome hg19. Unconvertible positions are marked red on the karyogram, annotated with HGNC symbol or ENSEMBL gene ID (if HGNC not available), retrieved from biomaRt_2.30.0.

Table 2. Number of samples from nine platforms in use case examples.

	all	1000 samples
CytoScanHD_Array	2963	73
CytoScan750K_Array	173	2
Mapping50K_Hind240	2699	58
Mapping50K_Xba240	3303	72
Mapping10K_Xba142	912	19
Mapping250K_Nsp	9738	223
Mapping250K_Sty	7561	184
GenomeWideSNP_6	16570	359
GenomeWideSNP_5	552	10

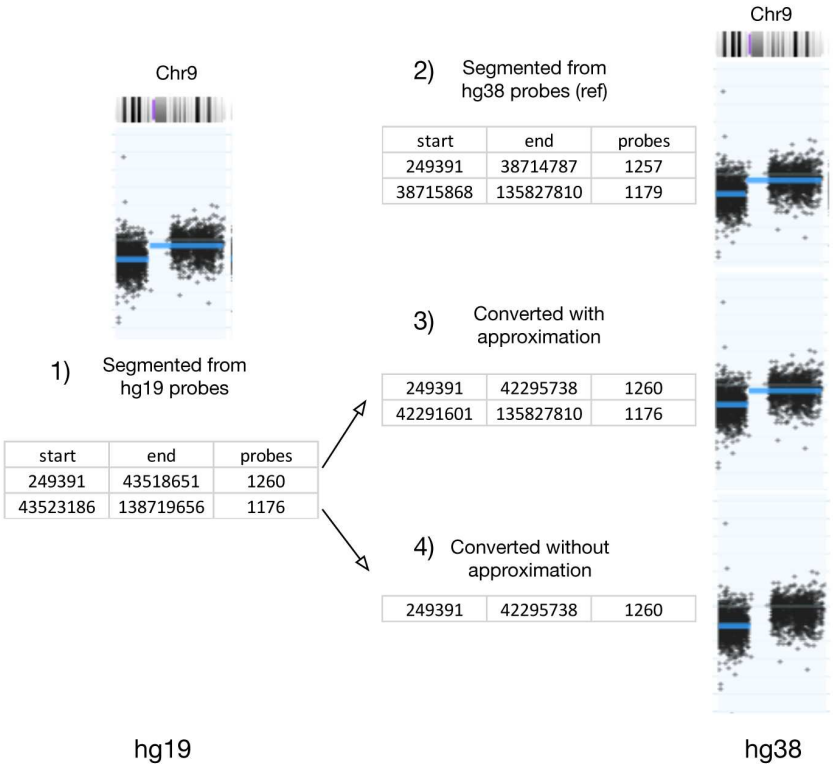

Figure 4. Chromosome 9 from GSM276858 with probe and segment data from hg19 coordinate(left) and hg38(right). Segments directly processed from hg38 probes are used as reference (top right). hg19 segments converted with approximation (middle right) and without approximation (bottom right) are used for comparison.

(3) segments in hg38, which were converted from hg19 to hg38 using *segment_liftover* with approximate conversion; (4) segments in hg38, which were converted from hg19 to hg38 using *segment_liftover* without approximate conversion. Strategy (1) is the standard procedure of copy number segment calling, and is used a the basic reference to compare with (2). Strategy (2) converts the raw data (probes) to hg38, then follows the same procedure as in (1). It is used as a reference for (3) and (4) to compare with. Strategies (3) and (4) convert segments (results from (1)) from hg19 to hg38, the difference is whether approximate conversion is used.

Table 3 shows the comparison of conversion results in average between (3), (4) and (2) (complete table in Supplementary Table 2). Exact segment matches are categorized as "perfect"; "minor difference" is defined the same as condition (3) of quality control; the rest of the segments are categorized as "significant difference"; "sum" is average number of segments per sample. By comparing the sum of *reference hg19*, *reference hg38* and *approximation*, it shows that the conversion result is very close to the result of the standard pipeline. The difference between converted and generated sums is much smaller than the difference between two generated sums of different genome

Table 3. Number of segments with or without approximation on average.

	perfect	minor difference	significant difference	sum
reference hg19	*na*	*na*	*na*	218.42
reference hg38	*na*	*na*	*na*	215.04
approximation	198.18	6.24	10.25	214.67
no approximation	198.18	5.23	10.01	213.42

versions. On average, the approximate conversion could rescue one additional segment per file. Finally, we zoom into a specific example on chromosome 9 in GSM276858 (illustrated in Figure 4). Because of the removal of probes from hg19 to hg38, the second segment will be lost without approximate conversion.

The two examples above demonstrated the efficiency and effectiveness of *segment_liftover* in processing large number of probe and segment files. It can provide conversion results that are similar to results generated from the standard pipeline. Moreover, with approximate conversion, the number of properly converted segments is slightly increased. In general, *segment_liftover* is able to provide reliable conversions and the ease of use.

Summary

Translation between genome versions of sequencing data is a tedious but crucial task in bioinformatics. With the functionalities of automated batching, approximate conversion and segment conversion, *segment_liftover* can dramatically reduce the complexity and workload of such data processing. Furthermore, *segment_liftover*'s detailed logs of execution result provide an easy and clear foundation for follow up analysis.

Software availability

1. pip version: https://pypi.python.org/pypi/segment-liftover
2. Latest source code: https://github.com/baudisgroup/segment-liftover
3. Archived source code as at time of publication: https://doi.org/10.5281/zenodo.1186803[7]
4. Software license: MIT

Competing interests

No competing interests were disclosed.

Grant information

BG is recipient of a grant from the China Scholarship Council (CSC 201606620032). QH is supported through the University of Zurich's "CanDoc" program.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgments

We thank Paula Carrio Cordo and the participants of the "Zurich Seminars in Bioinformatics" for helpful discussions.

Supplementary material

Supplementary Table 1 : genetic elements affected by the conversion The 38 genetic elements that are affected by the segment_liftover conversion from hg19 to hg38.

Click here to access the data.

Supplementary Table 2 : complete conversion results of different strategies A complete list of conversion results of comparing different conversion strategies using 1000 random samples.

Click here to access the data.

References

1. Kent WJ, Haussler D: **Assembly of the working draft of the human genome with GigAssembler.** *Genome Res.* 2001; **11**(9): 1541–1548.
PubMed Abstract | Publisher Full Text | Free Full Text
2. Kuhn RM, Haussler D, Kent WJ: **The UCSC genome browser and associated tools.** *Brief Bioinform.* 2013; **14**(2): 144–161.
PubMed Abstract | Publisher Full Text | Free Full Text
3. Zhao H, Sun Z, Wang J, *et al.*: **Crossmap: a versatile tool for coordinate conversion between genome assemblies.** *Bioinformatics.* 2014; **30**(7): 1006–1007.
PubMed Abstract | Publisher Full Text | Free Full Text
4. NCBI Resource Coordinators: **Database resources of the National Center for Biotechnology Information.** *Nucleic Acids Res.* 2016; **44**(D1): D7–D19.
PubMed Abstract | Publisher Full Text | Free Full Text
5. Cai H, Gupta S, Rath P, *et al.*: **arrayMap 2014: an updated cancer genome resource.** *Nucleic Acids Res.* 2015; **43**(Database issue): D825–D830.
PubMed Abstract | Publisher Full Text | Free Full Text
6. Olshen AB, Venkatraman ES, Lucito R, *et al.*: **Circular binary segmentation for the analysis of array-based DNA copy number data.** *Biostatistics.* 2004; **5**(4): 557–572.
PubMed Abstract | Publisher Full Text
7. Gao B, Baudis M: **baudisgroup/segment-liftover: First public version (Version 0.948).** *Zenodo.* 2018.
Data Source

Open Peer Review

Current Referee Status: ✔ ✔

Version 2

Referee Report 08 June 2018

doi:10.5256/f1000research.16527.r34839

✔ **Ryan K. Dale**
Laboratory of Cellular and Developmental Biology, National Institute of Diabetes and Digestive and Kidney Diseases, National Institutes of Health, Bethesda, MD, USA

The revisions address my concerns -- and the new demo mode is especially useful.

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Referee Report 17 April 2018

doi:10.5256/f1000research.15390.r32847

? **Ryan K. Dale**
Laboratory of Cellular and Developmental Biology, National Institute of Diabetes and Digestive and Kidney Diseases, National Institutes of Health, Bethesda, MD, USA

The authors describe how using existing genome coordinate conversion tools can be problematic on large domains in which continuity of the converted region is preferred over precision of the coordinate conversion. They describe a method that iteratively retries the conversion and applies QC criteria to the original and iterative conversions. The method is implemented using a Python package that wraps the UCSC liftover tool and performs the iterative remapping and QC steps, and provides this functionality over directories of files instead of on a per-file basis.

To make it scientifically sound there should be a clear example in the documentation that works (details below). I have additional concerns and suggestions, particularly about the code, that I hope will improve the paper and the tool.

Major comments:

- Figure 3 is difficult to interpret as-is because some probes are double-counted across bars. I think a

better approach would be to have a stacked bar for which the full height indicates the total number of probes that can be subset into converted and not-converted. These can be further subset by colors or by an adjacent stacked bar to indicate the proportion of converted that were directly converted or approximately converted, and the proportion of QC-failed.

- I see in the code that there are some human-specific hard-coded conversions between chr23/24 and chrX/Y. This may cause unexpected results when running on other genomes. One solution would be to run a check on the chromosomes observed in the chainfile to see if it looks like a human liftover before doing the 23/24 -> X/Y conversions.

- Currently the liftOver executable is required to be in the working directory or manually specified with the "--liftover" arg. A friendlier solution would be to expect liftOver to be found on the PATH variable (while also keeping the ability to explicitly specify if needed). This avoids having to symlink the executable to the working directory or requiring system-dependent command-line arguments.

- A "logs" directory is created in the working directory, even just after running "segment_liftover --help". It would be nice to defer creating this directory until it's actually needed. This also means that running parallel jobs on a cluster from the same directory will clobber the log files, making debugging difficult. It would be better to expose log directory configuration on the CLI and/or put the logs directory as a subdirectory of the output directory by default.

- the "tmp" directory behaves the same way (clutters the working directory; clobbers files when running in parallel). Furthermore it's not cleaned up after running. Better would be to use Python's tempfile module, which respects the environment's TMPDIR variable and is periodically automatically cleaned.

- Example data in the correct directory structure should be provided, and a single command to run that data should be provided. That way the user can easily verify that everything is installed correctly. As it is, I had to create the directory structure and paste the example segments content from the README into a new file in that directory. That allowed segment_liftover to run, but then it failed on the example segments data in the README. Digging around in the logs, in general.log there appears to be an issue with int vs str expectations when loading as pandas.DataFrame (this is with pandas v0.22.0).

- Are there requirements for chromosome names in segment files or probe files? The examples use integer chromosome names, but liftOver chainfiles use UCSC "chr" chromosome names. It appears that using an input file with "chr" names fails, again from expectations about int vs str columns in pandas. It would be good to describe what kind of chromosome names are supported (or link to a segments file specification if there is one).

Minor comment: Typo in text, "tensor" -> "tens or", and various typos in help text when running segment_liftover

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Reader Comment 01 Jun 2018

Bo Gao, University of Zurich, Switzerland

"- Figure 3 is difficult to interpret..."
We also discussed the best way to show this. Now it's replaced with a table.

"- I see in the code that there are some human-specific hard-coded conversions..."
Sorry for this overlook, the tool was originally developed for the human genome, and we forgot to modify this, now the hard coding replaced with generally coding. Following your recommendation, we investigated the feasibility of inferring valid chromosome names from the chainfile, but it turns out for some small genomes, the chainfile may not contain all valid chromosome names. In the end, as *UCSC liftOver* will ignore unrecognized chromosome names, we decide to pass the duty of providing correct chromosome names to the user.

" *- Currently the liftOver executable is required to be in the working directory...*"
Now the tool looks for the *liftOver* in the system path by default, and users can still manually specify with "-l".

" *- A "logs" directory is created in the working directory...*"
Modifications: (1) the log file and directory is created only after something is written to the file. (2) the logs/ directory is moved to the output directory by default. (3) users can also specify the log directory with a new option.

" *- the "tmp" directory behaves the same way...*"
The tmp/ directory is also moved to the output directory and will be properly removed after execution. Because sometimes we need to inspect the temp files, we decide not to use the Python module for easy accessing.

" *- Example data in the correct directory structure should be provided...*"
A new option "--demo" is added to provide a small demo dataset and a quick run.

" *- Are there requirements for chromosome names...*"
README and MANUAL have been updated to provide a better explanation of accepted format.

"*Typo in text...*"
All text has been revised.

Thank you very much for all the valuable suggestions!

Competing Interests: No competing interests were disclosed.

Referee Report 26 March 2018

doi:10.5256/f1000research.15390.r31947

Oscar Krijgsman
Division of Molecular Oncology and Immunology, Netherlands Cancer Institute, Amsterdam, Netherlands

This manuscript discusses a new tool developed by the authors that improves the 'liftover' between genomic builds. "Segment_liftover" is built on the already available liftover tool from UCSC with the novelty that it the tool implements a better conversion between genomic builds for stretches of the genome (segments) like in for example DNA copy number analyses. In addition, the tool provides options to perform the analysis in batches and provides log files of execution and results. The tool, as available on github, provides ample documentation and examples to run without experiencing too many difficulties.

The authors provide an interesting new tool that shows an improvement over existing tools. In general, the manuscript is clear and concise. However, a few minor points need to be addressed to improve the interpretation of the results described in the manuscript.

- The second use case is basically a direct comparison between UCSC liftover and the newly developed segment_liftover tool. However, this is not explicitly mentioned in the text. In the current version it is not evident that you did the nice/required comparison with a currently available tool. Furthermore, a little more explanation on the 4 different strategies would benefit the reader.

- Figure 3 does not convincingly convey its message. For example, it is not clear from the figure that the percentage of unconvertible segments is higher that probes although properly mentioned in the text. But also, the difference between the first bars is over a factor 1000. I understand the choice of a log10 scale but the authors might consider a different way of showing this data (table? Add percentages or number?).

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reader Comment 01 Jun 2018

Bo Gao, University of Zurich, Switzerland

" *The second use case is basically a direct comparison...*"
Our intention is to compare the performance of *segment_liftover* with the standard copy number segment calling procedure, which generates segments from probes. We have expanded and revised this part to provide more information about the settings and the purpose.

" Figure 3 does not convincingly convey its message..."
We also had discussions about the best way to show the information, now we have replaced the figure with a table for clarity.

Thank you for the suggestions!

Competing Interests: No competing interests were disclosed.